# Maze (Shortest Path)

Generated by Doxygen 1.12.0

# Chapter 1

# README

## 1.1 cosc2804-assignment3-template

### 1.1.1 Team Allocation

| | Member 1 (Tom Castanelli) [s4072172] | Member 2 (Liam Moore) [s4095280] | Member 3 (Erfan Samandarian) [s4089117] |
|---|---|---|---|
| Testing | Test to cover Member 1's and Member 2's work | Test to cover Member 2's and Member 3's work | Test to cover Member 3's and Member 1's work |
| Base Program | Maze Generation | Maze Solving | Build maze & Cleaning the world |
| Enhancements | E3 | E1 | E2 |

### 1.1.2 Team Video Link

`https://drive.google.com/file/d/1XGCHKvxCXyFYM-tzKQ7rC1j0PZlwhJqQ/view?usp=sharing`

### 1.1.3 Team Progress

#### 1.1.3.1 Toms's Commits

- Created basic Menu structure and navigation
- Created Maze class and populated with setters/getters, data etc.
- Implemented Test Mode check
- Implemented Import Maze from terminal
- Implemented Random Maze Generation
- Implemented Testing Mode for Maze Generation

#### 1.1.3.2 Liam's Commits

- Created test_invalidCharacterInput.input and expout files
- Created test_buildMaze.input and expout files
- Implemented functionality to find open blocks in maze to randomly teleport the player into
- Implemented findExitCoords() to get the exit coords of the maze for use in multiple functions
- Implemented maze solver using right-hand wall follower algorithm to show user the exit path
- Implemented testing mode for placing player in maze and solving maze using RHWF algorithm
- (E1) Implemented functionality to generate random mazes which take pre-existing terrain into account when generating their structure.
- (E1) Altered "Solve maze manually" and "Show Escape Route" functions to account for uneven terrain

#### 1.1.3.3 Erfan's Commits

- Created test_generateMaze.input and expout file
- Created Build_Agent to contain type int values
- Created Block_Array::Vector to contain type int values
- Refactored Block_Array::Vector to use template for general data types
- Refactored Save_Terrain() to use getBlocks for heap connection + speed
- Implemented breadth first search for enhancement (E2) in my fork
- Implemented test for find shortest path enhancement in the fork
- Changed Restore_Terarin() to clear maze floor first to prevent falling sand or gravel blocks

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 4

# Class Documentation

## 4.1 Agent Struct Reference

#### Public Member Functions

- **Agent** (mcpp::Coordinate startLoc)

The documentation for this struct was generated from the following files:

- Agent.h
- Agent.cpp

## 4.2 Block Struct Reference

```
#include <Block.h>
```

#### Public Attributes

- int **x** = DEFAULT_X
- int **y** = DEFAULT_Y
- int **z** = DEFAULT_Z
- mcpp::BlockType **block_type** = DEFAULT_BLOCK

### 4.2.1 Detailed Description

Represents a block with coordinates and type

The documentation for this struct was generated from the following file:

- Block.h

## 4.3 Maze Class Reference

### Public Member Functions

- **Maze** (unsigned int xLen, unsigned int zLen, bool mode)
- **Maze** (std::vector< std::vector< char > > mazeVec)
- void **Set_Len** (unsigned int xLen, unsigned int zLen)
- std::vector< unsigned int > **Get_Len** ()
- void **Set_Mode** (bool mode)
- bool **Get_Mode** ()
- void **Set_Maze** (std::vector< std::vector< char > > mazeVec)
- std::vector< std::vector< char > > **Get_Maze** ()
- void **Generate** ()
- void **Print** ()
- void **Print** (std::vector< std::vector< char > > mazeVec)
- unsigned int **getXLen** ()
- unsigned int **getZLen** ()

The documentation for this class was generated from the following files:

- Maze.h
- Maze.cpp

## 4.4 Place_Maze Class Reference

```
#include <Place_Maze.h>
```

### Public Member Functions

- bool Has_Terrain ()
- void Build_Maze ()
- void Flatten_Terrain ()
- void Load_Maze (std::vector< std::vector< char > > maze_vec, int length, int width)
- void Restore_Terrain ()
- void Save_Terrain ()
- mcpp::Coordinate Get_Player_Build_Pos ()
- void Set_Player_Build_Pos (mcpp::Coordinate player_pos)
- void Clear_Maze ()
- void Print_Maze_Size ()
- void Place_Carpet ()

### 4.4.1 Detailed Description

Manages the placement and restoration of a maze

### 4.4.2 Member Function Documentation

#### 4.4.2.1 Build_Maze()

```
void Place_Maze::Build_Maze ()
```

Builds the maze

#### 4.4.2.2 Clear_Maze()

```
void Place_Maze::Clear_Maze ()
```

Clears the maze

#### 4.4.2.3 Flatten_Terrain()

```
void Place_Maze::Flatten_Terrain ()
```

Flattens the terrain

#### 4.4.2.4 Get_Player_Build_Pos()

```
mcpp::Coordinate Place_Maze::Get_Player_Build_Pos ()
```

Gets the player's build position

#### 4.4.2.5 Has_Terrain()

```
bool Place_Maze::Has_Terrain ()
```

Checks if terrain is available

#### 4.4.2.6 Load_Maze()

```
void Place_Maze::Load_Maze (
             std::vector< std::vector< char > > maze_vec,
             int length,
             int width)
```

Loads the maze from a vector

#### 4.4.2.7 Place_Carpet()

```
void Place_Maze::Place_Carpet ()
```

Places carpet in the maze

#### 4.4.2.8 Print_Maze_Size()

```
void Place_Maze::Print_Maze_Size ()
```

Prints the maze size

#### 4.4.2.9 Restore_Terrain()

```
void Place_Maze::Restore_Terrain ()
```

Restores the original terrain

#### 4.4.2.10 Save_Terrain()

```
void Place_Maze::Save_Terrain ()
```

Saves the current terrain

#### 4.4.2.11 Set_Player_Build_Pos()

```
void Place_Maze::Set_Player_Build_Pos (
            mcpp::Coordinate player_pos)
```

Sets the player's build position

The documentation for this class was generated from the following files:

- Place_Maze.h
- Place_Maze.cpp

## 4.5 Solve_Maze Class Reference

### Static Public Member Functions

- static bool **isOutsideMaze** (int &, const std::vector< mcpp::Coordinate > &)
- static mcpp::Coordinate **findExitCoords** (int, int, mcpp::Coordinate)
- static bool **placePlayerInMaze** (const std::vector< std::vector< char > > &, mcpp::Coordinate, int, int, bool)
- static bool **showEscapeRoute** (int, int, mcpp::Coordinate, const std::vector< std::vector< char > > &, bool)
- static void **getOpenBlocks** (const std::vector< std::vector< char > > &, mcpp::Coordinate, int, int)

The documentation for this class was generated from the following files:

- Solve_Maze.h
- Solve_Maze.cpp

## 4.6 Terrain_Array Class Reference

```
#include <Terrain_Array.h>
```

### Public Member Functions

- **Terrain_Array** (const Terrain_Array &)=delete
- Terrain_Array & **operator=** (const Terrain_Array &)=delete
- void add_block (int x, int y, int z, mcpp::BlockType block)
- Block get_block (int index) const
- int size () const
- Block ∗ **begin** () const
- Block ∗ **end** () const
- void print () const
- void clear ()

### 4.6.1 Detailed Description

Manages a dynamic array of Blocks

### 4.6.2 Member Function Documentation

#### 4.6.2.1 add_block()

```
void Terrain_Array::add_block (
             int x,
             int y,
             int z,
             mcpp::BlockType block)
```

Adds a block at the specified coordinates

#### 4.6.2.2 clear()

```
void Terrain_Array::clear ()
```

Clears all blocks

#### 4.6.2.3 get_block()

```
Block Terrain_Array::get_block (
             int index) const
```

Retrieves a block by index

#### 4.6.2.4 print()

```
void Terrain_Array::print () const
```

Prints all blocks

#### 4.6.2.5 size()

```
int Terrain_Array::size () const
```

Returns the number of blocks

The documentation for this class was generated from the following files:

- Terrain_Array.h
- Terrain_Array.cpp

# Chapter 5

# File Documentation

## 5.1 Agent.h

```
#ifndef AGENT_H
#define AGENT_H

#include <mcpp/mcpp.h>

#define MOVE_XPLUS mcpp::Coordinate(1, 0, 0)
#define MOVE_XMINUS mcpp::Coordinate(-1, 0, 0)
#define MOVE_ZPLUS mcpp::Coordinate(0, 0, 1)
#define MOVE_ZMINUS mcpp::Coordinate(0, 0, -1)

enum solveAlgorithm {
    RIGHT_HAND_FOLLOW,
    BREATH_FIRST_SEARCH,
};

enum AgentOrientation { X_PLUS, Z_PLUS, X_MINUS, Z_MINUS };

struct Agent {
    Agent(mcpp::Coordinate startLoc);
    ~Agent();
};

#endif // AGENT_H
```

## 5.2 Block.h

```
#ifndef BLOCK_ARRAY_H
#define BLOCK_ARRAY_H

#include <mcpp/block.h>

#define DEFAULT_X 0
#define DEFAULT_Y 0
#define DEFAULT_Z 0
#define DEFAULT_BLOCK mcpp::Blocks::AIR

struct Block {
    int x = DEFAULT_X;
    int y = DEFAULT_Y;
    int z = DEFAULT_Z;
    mcpp::BlockType block_type = DEFAULT_BLOCK;
};

#endif // BLOCK_ARRAY_H
```

## 5.3 Maze.h

```
#ifndef ASSIGN_MAZE_H
#define ASSIGN_MAZE_H
```

```

#include <mcpp/mcpp.h>
#include <vector>

// Maze class stores only information related to construction or generation of
// the maze, not the maze’s location in minecraft
class Maze {
  public:
    // Constructors
    Maze(unsigned int xLen, unsigned int zLen, bool mode);
    Maze(std::vector<std::vector<char» mazeVec);
    Maze();

    void Set_Len(unsigned int xLen, unsigned int zLen);
    std::vector<unsigned int> Get_Len();

    void Set_Mode(bool mode);
    bool Get_Mode();

    void Set_Maze(std::vector<std::vector<char» mazeVec);
    std::vector<std::vector<char» Get_Maze();

    void Generate();

    void Print();
    void Print(std::vector<std::vector<char» mazeVec);

    unsigned int getXLen();
    unsigned int getZLen();

    // Destructors
    ~Maze();

  private:
    // xLen = columns
    unsigned int xLen;
    // zLen = rows
    unsigned int zLen;
    bool mode;
    std::vector<std::vector<char» mazeVec;

    // This Function checks adjacent squares for a specific character
    void Check_Adj(std::vector<std::vector<char»& mazeVec, unsigned int& xCurr,
                   unsigned int& zCurr, char targetChar, unsigned int dist,
                   bool& north, bool& east, bool& south, bool& west);
};

#endif
// ASSIGN_MAZE_H
```

## 5.4 menuUtils.h

```
#include <iostream>

//TODO: Move functions to menuUtils.cpp
void printStartText(void){
    std::cout « std::endl;
    std::cout « "Welcome to MineCraft MazeRunner!" « std::endl;
    std::cout « "--------------------------------" « std::endl;
}

void printMainMenu(void){
    std::cout « std::endl;
    std::cout « "------------- MAIN MENU -------------" « std::endl;
    std::cout « "1) Generate Maze" « std::endl;
    std::cout « "2) Build Maze in MineCraft" « std::endl;
    std::cout « "3) Solve Maze" « std::endl;
    std::cout « "4) Show Team Information" « std::endl;
    std::cout « "5) Exit" « std::endl;
    std::cout « std::endl;
    std::cout « "Enter Menu item to continue: " « std::endl;
}

void printGenerateMazeMenu(void){
    std::cout « std::endl;
    std::cout « "------------- GENERATE MAZE -------------" « std::endl;
    std::cout « "1) Read Maze from terminal" « std::endl;
    std::cout « "2) Generate Random Maze" « std::endl;
    std::cout « "3) Back" « std::endl;
    std::cout « std::endl;
    std::cout « "Enter Menu item to continue: " « std::endl;
}

```

```
void printGeneratePrompt(void){
    std::cout « "In Minecraft, navigate to where you need the maze" « std::endl « "to be built in Minecraft and type - done:";
    std::cout « std::endl;
}


void printEnterLW(void){
    std::cout «"Enter the length and width of maze:";
    std::cout « std::endl;
}

void printEnterStruct(void){
    std::cout «"Enter the maze structure:";
    std::cout « std::endl;
}

void printMazeReadSuccess(void) {
    std::cout «"Maze read successfully";
    std::cout « std::endl;
}

void printPrintingStart(void) {
    std::cout « "**Printing Maze**";
    std::cout « std::endl;
}

void printPrintingEnd(void){
    std::cout «"**End Printing Maze**";
    std::cout « std::endl;
}

void printReadyToSolve(void){
    std::cout «"Maze ready to Solve...";
    std::cout « std::endl;
}

void printSolveMazeMenu(void){
    std::cout « std::endl;
    std::cout « "------------- SOLVE MAZE -------------" « std::endl;
    std::cout « "1) Solve Manually" « std::endl;
    std::cout « "2) Show Escape Route" « std::endl;
    std::cout « "3) Back" « std::endl;
    std::cout « std::endl;
    std::cout « "Enter Menu item to continue: " « std::endl;
}


void printTeamInfo(void){
    std::cout « std::endl;
    std::cout « "Team members:" « std::endl;
    std::cout « "\t [1] Erfan Samandarian (s4089117@student.rmit.edu.au)" « std::endl;
    std::cout « "\t [2] Liam Moore (s4095280@student.rmit.edu.au)"        « std::endl;
    std::cout « "\t [3] Thomas Castanelli (s4072172@student.rmit.edu.au)" « std::endl;
    std::cout « std::endl;
}

void printExitMessage(void){
    std::cout « std::endl;
    std::cout « "The End!" « std::endl;
    std::cout « std::endl;
}

void printErrorMessageException(void){
    std::cout « "Error: Unknown Exception";
    std::cout « std::endl;
}

void printErrorMessageInput(std::string errorStr) {
    std::cout « "Input Error: " « errorStr;
    std::cout « std::endl;
}

void printErrorMessageRange(int in1, int in2){
    std::cout « "Input Error: Enter a number between ";
    std::cout « in1;
    std::cout « " and ";
    std::cout « in2;
    std::cout « " ....";
    std::cout « std::endl;
}

// error if entered length and width (i.e. 3 and 5) are different to entered maze dimensions.
void printErrorMessageArraySizeIncorrect(int length, int width) {
    std::cout « "Input Error: Entered length (" « length « ") and width (" « width « ") differs from manually entered maze’s dimensions";
    std::cout « std::endl;
```

```
}

void printErrorMessageNoMaze() {
    std::cout « "Input Error: No available maze to solve";
    std::cout « std::endl;
}

void printErrorMessageInvalidMazeSize() {
    std::cout « "Input Error: At least one maze dimension has not been entered";
    std::cout « std::endl;
}

void printErrorPlayerNotInMaze() {
    std::cout « "Input Error: Player is not in maze";
    std::cout « std::endl;
}

// probably won't ever happen but eh
void printErrorMazeNoSpace() {
    std::cout « "Input Error: Maze has no open space";
    std::cout « std::endl;
}

void printBasePoint(mcpp::Coordinate playerPos) {
    std::cout « "BasePoint: " « playerPos « std::endl;
}
```

## 5.5 Place_Maze.h

```
#ifndef PLACE_MAZE_H
#define PLACE_MAZE_H

#include "Solve_Maze.h"
#include "Terrain_Array.h"

#include <mcpp/mcpp.h>

#include <algorithm>
#include <chrono>
#include <iostream>
#include <iterator>
#include <thread>
#include <vector>

const std::vector<mcpp::BlockType> odd_blocks = {mcpp::Blocks::AIR,
                                                 mcpp::Blocks::STILL_WATER,
                                                 mcpp::Blocks::STILL_LAVA,
                                                 mcpp::Blocks::FLOWING_WATER,
                                                 mcpp::Blocks::FLOWING_LAVA,
                                                 mcpp::Blocks::LILY_PAD,
                                                 mcpp::Blocks::TALL_GRASS,
                                                 mcpp::Blocks::ACACIA_LEAVES,
                                                 mcpp::Blocks::ACACIA_SAPLING,
                                                 mcpp::Blocks::ACTIVATOR_RAIL,
                                                 mcpp::Blocks::BIRCH_LEAVES,
                                                 mcpp::Blocks::BIRCH_SAPLING,
                                                 mcpp::Blocks::CAKE_BLOCK,
                                                 mcpp::Blocks::COBWEB,
                                                 mcpp::Blocks::DARK_OAK_LEAVES,
                                                 mcpp::Blocks::DARK_OAK_SAPLING,
                                                 mcpp::Blocks::DEAD_BUSH,
                                                 mcpp::Blocks::DEAD_SHRUB,
                                                 mcpp::Blocks::DETECTOR_RAIL,
                                                 mcpp::Blocks::FLOWER_POT,
                                                 mcpp::Blocks::GRASS_PATH,
                                                 mcpp::Blocks::JUNGLE_LEAVES,
                                                 mcpp::Blocks::JUNGLE_SAPLING,
                                                 mcpp::Blocks::LARGE_FERN,
                                                 mcpp::Blocks::LILAC,
                                                 mcpp::Blocks::OAK_LEAVES,
                                                 mcpp::Blocks::OAK_SAPLING,
                                                 mcpp::Blocks::POWERED_RAIL,
                                                 mcpp::Blocks::RAIL,
                                                 mcpp::Blocks::ROSE_BUSH,
                                                 mcpp::Blocks::SPRUCE_LEAVES,
                                                 mcpp::Blocks::SUNFLOWER};

class Place_Maze {
  public:
    Place_Maze() {};

    bool Has_Terrain();
    void Build_Maze();
    void Flatten_Terrain();
    void Load_Maze(std::vector<std::vector<char» maze_vec, int length,
                   int width);
    void Restore_Terrain();
    void Save_Terrain();

    mcpp::Coordinate Get_Player_Build_Pos();
    void Set_Player_Build_Pos(mcpp::Coordinate player_pos);

    void Clear_Maze();
    void Print_Maze_Size();

    void Place_Carpet();

  private:
    mcpp::MinecraftConnection mc;
    mcpp::Coordinate player_position;

    int length = 0;
    int width = 0;

    Terrain_Array terrain;
    Terrain_Array maze;

    mcpp::BlockType player_position_block;

    /* Blocks for placement of carpet */
    mcpp::BlockType carpet_block;
    mcpp::BlockType one_below_carpet_block;
    mcpp::BlockType two_below_carpet_block;
    mcpp::BlockType one_above_carpet_block;
    mcpp::BlockType two_above_carpet_block;

    /* Coordinates for placement of carpet */
    mcpp::Coordinate carpet_coord;
    mcpp::Coordinate one_below_carpet_coord;
    mcpp::Coordinate two_below_carpet_coord;
    mcpp::Coordinate one_above_carpet_coord;
    mcpp::Coordinate two_above_carpet_coord;
};

#endif
```

## 5.6 Solve_Maze.h

```
#ifndef SOLVE_MAZE_H
#define SOLVE_MAZE_H

#include <chrono>
#include <iostream>
#include <mcpp/mcpp.h>
#include <thread>

class Solve_Maze {
  public:
    // returns true if solver reaches exit, and stops solver
    static bool isOutsideMaze(int&, const std::vector<mcpp::Coordinate>&);

    // used to find the coordinates of the exit using the length and width of
    // the maze and checking for openings on the edges
    static mcpp::Coordinate findExitCoords(int, int, mcpp::Coordinate);

    static bool placePlayerInMaze(const std::vector<std::vector<char»&,
                                  mcpp::Coordinate, int, int, bool);

    // option [2], guides the player out of the maze
    static bool showEscapeRoute(int, int, mcpp::Coordinate,
                                const std::vector<std::vector<char»&, bool);

    // populated openCoordsField with open blocks within the maze
    static void getOpenBlocks(const std::vector<std::vector<char»&,
                              mcpp::Coordinate, int, int);

  private:
    // fields
    static mcpp::Coordinate exitCoords;
    static std::vector<mcpp::Coordinate> openCoordsField;
};

#endif // SOLVE_MAZE_H
```

## 5.7 Terrain_Array.h

```
#ifndef TERRAIN_ARRAY_H
#define TERRAIN_ARRAY_H

#include "Block.h"

#include <iostream>
#include <stdexcept>

class Terrain_Array {
  public:
    Terrain_Array();
    ~Terrain_Array();

    Terrain_Array(const Terrain_Array&) = delete;
    Terrain_Array& operator=(const Terrain_Array&) = delete;

    void add_block(int x, int y, int z, mcpp::BlockType block);
    Block get_block(int index) const;
    int size() const;

    Block* begin() const { return blocks; }
    Block* end() const { return blocks + block_count; }

    void print() const;

    void clear();

  private:
    void resize();
    Block* blocks;
    int block_count;
    int capacity;
};

#endif // TERRAIN_ARRAY_H
```

# Index